Attorney Docket No. FULCP006

PATENT APPLICATION

ASYNCHRONOUS MULTIPLE-ORDER ISSUE SYSTEM ARCHITECTURE

Inventors: Andrew Lines of
Calabasas, California
United States citizen

Robert Southworth of
Pasadena, California
United States citizen

Uri Cummings of
Oak Park, California
United States citizen

Assignee: Fulcrum Microsystems Inc., a
California corporation

BEYER WEAVER & THOMAS, LLP
P.O. Box 778
Berkeley, California 94704-0778
(510) 843-6200

PATENT

Attorney Docket No. FULCP006

ASYNCHRONOUS MULTIPLE-ORDER ISSUE SYSTEM ARCHITECTURE

RELATED APPLICATION DATA

[0001] The present application claims priority from U.S. Provisional Patent Application 60/411,717 for ASYNCHRONOUS MULTIPLE-ORDER ISSUE PROCESSOR ARCHITECTURE filed on September 16, 2002, the entire disclosure of which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

[0002] The present invention relates to asynchronous system architectures with orders greater than single-issue.

[0003] The goal of a dual issue system in a processor architecture is to be able to execute two instructions per cycle. This is accomplished with two parallel pipelines for handling the tasks for each of a pair of instructions. In the standard synchronous design, a pair of instructions are simultaneously received each clock cycle. Because of the interdependency between instructions and the limitation that the instructions be issued simultaneously (i.e., at only one time in each clock cycle), there are inherent system complexities which prevent system performance from approaching the desired goal. That is, when the instructions are issued simultaneously, there are issues relating to program order and data dependency which require complex schemes for determining, for example, whether two instructions may be executed at the same time. These complex schemes not only hamper system performance, but make multiple issue architectures difficult to verify.

[0004] It is therefore desirable to provide circuits and techniques by which multiple and mixed-order-issue system architectures may be more readily implemented.

SUMMARY OF THE INVENTION

[0005] According to the present invention, an asynchronous system architecture is provided which benefits from the performance advantage of dual and higher-order issue resources without the logical complexity associated with the simultaneous issuance of potentially interdependent data. According to a specific embodiment, an asynchronous circuit is provided for processing units of data having a program order associated therewith. The circuit includes an N-way-issue resource comprising N parallel pipelines. Each pipeline is operable to transmit a subset of the units of data in a first-in-first-out manner. The asynchronous circuit is operable to sequentially control transmission of the units of data in the pipelines such that the program order is maintained. According to a more specific embodiment, an M-way-issue resource is provided along with interface circuitry operable to facilitate communication between the N-way-issue resource and the M-way-issue resource.

[0006] According to some embodiments, the interface circuitry is operable to facilitate transmission of selected ones of the data units from the N-way-issue resource to the M-way-issue resource. According to some embodiments, the interface circuitry is operable to facilitate transmission of selected ones of the data units from the M-way-issue resource to the N-way-issue resource. According to some embodiments, the interface circuitry is operable to facilitate transmission of first selected ones of the data units from the N-way-issue resource to the M-way-issue resource, and second selected ones of the data units from the M-way-issue resource to the N-way-issue resource.

[0007] According to another embodiment, a heterogeneous system is provided for processing units of data having a program order associated therewith. The system includes an N-way issue resource and at least one multiple-issue resource having an order different from N. The system also includes interface circuitry operable to facilitate communication between the N-way-issue resource and the at least one multiple-issue resource and to preserve the program order in all of the resources. According to a more specific embodiment, the at least one multiple-issue resource comprises a plurality of multiple-issue resources having different orders.

[0008] A further understanding of the nature and advantages of the present invention may be realized by reference to the remaining portions of the specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Fig. 1 is a block diagram illustrating a specific embodiment of the invention.

[0010] Fig. 2 is a block diagram illustrating a more specific embodiment of the invention.

[0011] Fig. 3 is a block diagram illustrating yet another embodiment of the invention.

[0012] Figs. 4A-4C are block diagrams illustrating various embodiments of the invention.

[0013] Fig. 5 is an exemplary representation of more general embodiment of the invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

[0014] Reference will now be made in detail to specific embodiments of the invention including the best modes contemplated by the inventors for carrying out the invention. Examples of these specific embodiments are illustrated in the accompanying drawings. While the invention is described in conjunction with these specific embodiments, it will be understood that it is not intended to limit the invention to the described embodiments. On the contrary, it is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. The present invention may be practiced without some or all of these specific details. In addition, well known features may not have been described in detail to avoid unnecessarily obscuring the invention.

[0015] It should also be noted that specific embodiments of the invention may be implemented according to a particular design style relating to quasi-delay-insensitive asynchronous VLSI circuits. However it will be understood that many of the principles and techniques of the invention may be used in other contexts such as, for example, non-delay insensitive asynchronous VLSI as well as synchronous VLSI. As such, the details of the exemplary design style referred to herein or any circuits or systems designed according to such a design style should not be construed as limiting the present invention.

[0016] Specific embodiments of the invention will now be described in the context of a particular asynchronous design style which is characterized by the storing of data in channels instead of registers. Such channels implement a FIFO (first-in-first-out) transfer of data from a sending circuit to a receiving circuit. As will become apparent, this automatic queuing of data in these channels is advantageously employed according to the invention.

[0017] Data wires run from the sender to the receiver, and an enable (i.e., an inverted sense of an acknowledge) wire goes backward for flow control. A four-phase handshake between neighboring circuits (processes) implements a channel. The four phases are in order: 1) Sender waits for high enable, then sets data valid; 2) Receiver waits for valid data, then lowers enable; 3) Sender waits for low enable, then sets data neutral; and 4) Receiver waits for neutral data, then raises enable. It should be noted that the use of this design style and this handshake protocol is for illustrative purposes and that therefore the scope of the invention should not be so limited.

[0018] According to other aspects of this design style, data are encoded using 1*ofN* encoding or so-called "one hot encoding." This is a well known convention of selecting one of N+1 states with N wires. The channel is in its neutral state when all the wires are inactive. When the kth wire is active and all others are inactive, the channel is in its kth state. It is an error condition for more than one wire to be active at any given time. For example, in certain embodiments, the encoding of data is dual rail, also called 1*of*2. In this encoding, 2 wires (rails) are used to represent 2 valid states and a neutral state. According to other embodiments, larger integers are encoded by more wires, as in a 1*of*3 or 1*of*4 code. For much larger numbers, multiple 1*ofN*'s may be used together with different numerical significance. For example, 32 bits can be represented by 32 *1of*2 codes or 16 1*of*4 codes.

[0019] According to other aspects of this design style, the design includes a collection of basic leaf cell components organized hierarchically. The leaf cells are the smallest components that operate on the data sent using the above asynchronous handshaking style and are based upon a set of design templates designed to have low latency and high throughput. Examples of such leaf cells are described in detail in "Pipelined Asynchronous Circuits" by A.M. Lines, *Caltech Computer Science Technical Report* CS-TR-95-21, Caltech, 1995, the entire disclosure of which is incorporated herein by reference for all purposes.

[0020] In some cases, the above-mentioned asynchronous design style may employ the language CSP (concurrent sequential processes) to describe high-level algorithms and circuit behavior. CSP is typically used in parallel programming software projects and in delay-insensitive VLSI. Applied to hardware processes, CSP is sometimes known as CHP (for Communicating Hardware Processes). For a description of this language, please refer to

"Synthesis of Asynchronous VLSI Circuits," by A.J. Martin, DARPA Order number 6202. 1991, the entirety of which is incorporated herein by reference for all purposes.

[0021] The transformation of CSP specifications to transistor level implementations for use with various techniques described herein may be achieved according to the techniques described in "Pipelined Asynchronous Circuits" by A.M. Lines, incorporated herein by reference above. However, it should be understood that any of a wide variety of asynchronous design techniques may also be used for this purpose.

[0022] The present invention provides an asynchronous multiple-issue architecture in which the issue of instructions in N parallel instruction pipelines are staggered in time, e.g., every half cycle in a dual-issue embodiment, such that N instructions are issued each cycle, but, due to the alternating (and therefore sequential) issuance, the interdependency between closely spaced instructions do not require the complexities or suffer from the resulting underperformance encountered in the typical multiple-issue synchronous system. Embodiments of the present invention take advantage of the "first-in-first-out" nature of asynchronous channels to preserve the logical program order of multiple instructions which are being executed in parallel pipelines.

[0023] In some ways, this approach may be conceptualized as a single issue system in which instructions are issued, on average, every 1/N cycles according to the asynchronous flow control. That is, the instruction pipeline hardware itself is N-way-issue, but from the perspective of architectural modeling and verification it may be thought of as a single issue system which reads N alternate instruction channels. However, because instructions are issued every 1/N cycles, when consecutive related instructions are issued, on the micro-architectural level, certain computations may need to be accomplished in 1/N cycles rather than a full cycle.

[0024] As will be seen, one of the advantages of this approach is that it allows a heterogeneous system in which an N-way-issue instruction pipeline interacts with different issue resources relatively seamlessly, simply by sequentially selecting the relevant channels. And due to the FIFO nature of the asynchronous channels, conversion back and forth between different order resources may be achieved while maintaining program order. In such a system, specific resources may be implemented according to the particular order architecture which is most suitable for that resource. Thus, for example, many of the architectural simplicities of single-issue resources may be retained in a system which operates with dual or multiple-issue performance.

[0025] According to a specific embodiment illustrated in Fig. 1, an asynchronous system 100 includes instances of a dual filter 102 for interfacing a dual issue instruction pipeline 104 with single issue resources (e.g., resource 106), and a "split" circuit 108 and two optional assign circuits 109 for going back from the single issue resource 106 to the dual issue pipeline 104. The parallel instruction pipelines have input channels IN[0] and IN[1], and output channels OUT[0] and OUT[1]. State loops 110 between the two pipelines enable the communication of dependencies between the pipelines. The latency requirement for a state going into and then exiting such a loop is half of a cycle.

[0026] During system operation, instructions alternately enter the parallel pipelines at a rate of two per cycle. Each instruction has an associated bit or other implicit information or characteristics which indicates whether the instruction needs to be forwarded on to the single issue resource, i.e., a request for access to the single issue resource. Instructions intended for the single issue resource are forwarded sequentially in the order received by the dual filter to the single issue resource. The output of the single issue resource are then forwarded back to the appropriate pipeline via the split and optional assign circuitry according to control information generated from the instructions themselves by dual request circuitry.

[0027] In the embodiment shown, the dual filter requires some mechanism to "filter" or remove data from the dual-issue stream which do not need to go to the single-issue resource. Otherwise, the single-issue resource would present a bottleneck which would detract from the performance of the dual-issue resource. The dual request circuit uses the same control information in a similar manner to facilitate injection of data from the single-issue resource back into the dual-issue stream in way which preserves program order.

[0028] The manner in which the N-way to M-way filters are implemented and the filtering decisions are made depends on the types of resources involved. For example, if the asynchronous system being implemented is a processor, the simplicity of the program counter and the manner in which it is normally incremented is such that implementation as a dual-issue resource may be desirable. However, because branching in processors is computationally expensive and is done relatively seldom, e.g., one in seven instructions, it may be desirable to implement the resource which calculates branch addresses as a single-issue resource. In such a case, the dual filter which provides the interface between the program counter and the branch address calculator would look at its copy of the instructions going through the dual-issue stream and remove any instructions which are not branch instructions.

[0029] According to a specific embodiment, dual filter 102 has two n-bit asynchronous input channels that correspond to the dual-issue input datapath, IN[0] and IN[1], and two 1-bit asynchronous input channels that together form the dual-issue input control. One n-bit asynchronous output channel is the single-issue output. The input control channel delivers a valid bit for the datapath channel. If the control value is true, then those data are passed to the output single-issue channel. If the control value is false, the data are consumed without being passed on and the single-issue output channel is not exercised. The input "issues" are read alternately.

[0030] According to a specific embodiment, dual request 112 has two 1-bit asynchronous input channels that together form the dual-issue input control, and one 1-bit asynchronous output channel that is single issue. The input control channel is a copy of the control channel used in dual filter 102. The output channel contains an ordered single-issue stream that encodes to which issue, e.g. odd or even, the corresponding data token belongs. Dual request 112 works much like dual filter 102 except that, instead of passing a datapath word, it simply notes from which pipeline the datapath word comes. The output of dual request 112 controls a split process 108. Each of the two outputs of split 108 is combined via optional assign circuits 109 with one of the two issues of the dual-issue datapath to facilitate the data re-entering the dual-issue datapath.

[0031] According to a specific embodiment, split process or circuit 108 is a 1 to 2 bus which reads a control channel, reads one token of input data from a single channel, then sends the data to one of 2 output channels selected by the value read from the control channel. For additional detail, see "Pipelined Asynchronous Circuits" by A. Lines incorporated by reference above.

[0032] According to various embodiments, re-entering into the dual-issue datapath can be accomplished in a number of ways which are computation specific. For example, it may be done with an optional assign (as shown), a merge, or a similar process. An optional assign process is a process with two datapath inputs X and Y, one datapath output Z, and a control input C. If C = 0 then data are received on X and sent to Z. If C = 1 then data are received on both channels X and Y, the X data are discarded and the Y data are sent to Z. In the pipelines of Fig. 1, two optional assign processes 109 are in the dual issue datapath, with the X input connected to upstream dual-issue pipes, and the Y input connected to the outputs of the split process in the conversion region from single to dual issue. The conversion region may employ copies and optional assigns to reduce the number of pipelines associated with the lower order resource.

[0033] According to a specific embodiment, the control input to the optional assigns is a copy of the dual-issue control of the dual-filter. The even control channel controls the even optional assign and the odd control channel controls the odd optional assign. Thus, if a data token did not get filtered out by the dual filter, then a corresponding token must return to the dual-issue datapath, i.e., the corresponding token will appear on channel Y of the optional assign.

[0034] According to another embodiment in which a Fetch for a RISC processor is implemented, the data produced by the single issue resource may not be used, i.e., data are filtered once at the dual-filter, and then potentially filtered again at the optional assign. According to this embodiment, additional control information from the dual-issue resource is employed to further filter the data from the single issue resource. According to one embodiment, an enhanced optional assign is provided which allows for a 3rd control state in which X and Y are read, X is passed to Z, and Y is discarded. Such an embodiment is useful, for example, for speculation.

[0035] According to a particular implementation, a merge process or circuit for use with the system of Fig. 1 is a 2 to 1 bus which reads a control channel, then reads a token of data from one of the input channels as selected by the value read from the control channel, then sends that data to the single output channel. See "Pipelined Asynchronous Circuits" by A. Lines incorporated by reference above. Merge processes may be used, for example, where there is a one-to-one mapping of data in the pipes of the dual-issue resource to the data in the pipes of the single-issue resource.

[0036] Another useful process which may be employed by various embodiments of the invention is a dual repeat process which has one single-issue data input channel, one dual-issue control input channel, and one dual-issue data output channel. In this process the control bit has 2 operations. The first is to send the data to the output issue (same issue number as the control bit), but not operate the input channel so the value is available for future use. The second is to send the data to the output issue (same issue number as the control bit), and remove the data from the input channel, so the next data on the input channel appears on the input when servicing the next issue. According to a specific embodiment, operation 1 must occur most of the time to preserve dual-issue performance.

[0037] Using such an interface as shown in Fig. 1, very complex processes which aren't executed all that frequently may be implemented as single issue resources in an otherwise dual issue architecture. If a significant number of system resources may be implemented this way, dual issue performance may be achieved in a circuit having closer to single issue area (and power) requirements. Thus, a microprocessor architecture may be implemented in which only some system resource are implemented as dual issue resources. For example, implementing a branching unit as dual issue can be very complex due, at least in part, to the fact that it necessitates computation of two program counters (PCs) at once. However, using the interface of the present invention allows such a branching unit to be implemented as a single issue resource, the target instruction generated by the branching unit being passed back to a dual issue fetch unit which continues to increment the single PC until there is another branch.

[0038] It should be noted that the system and circuits of Fig. 1 are merely exemplary and that many variations of the basic ideas illustrated are within the scope of the invention. That is, for example, the figure and the foregoing description show an implementation in which a dual-issue resource, i.e., the instruction pipeline, utilizes a single-issue resource in such a way that there is a one-to-one correspondence between the data exiting and returning to the dual-issue resource. It will be understood, however, that other implementations are contemplated in which this is not the case. For example, other embodiments may include ones in which data are not returned to the dual-issue resource, or, alternatively, in which the single-issue resource acts as a data source for the dual-issue resource, i.e., providing data to but not receiving data from the dual-issue resource. Additionally, data may be returned from the single-issue resource to the dual-issue resource in a more complex manner than one-to-one with the appropriate logic and flow control.

[0039] Referring to Fig. 2, an exemplary processor architecture 200 is provided in which dual issue is used between the instruction dispatch 202 and the register file 204, the icache 206 and the dispatch, the branch predict 208 and the icache, the fetch 210 and the branch predict, the dispatch and the fetch, and the dispatch and the writeback 212. On the other hand, the complicated instruction decoding circuitry 214 going into the icache is single issue. The execution pipelines 216 are single issue. The branch circuitry 218 is single issue.

[0040] In addition, certain portions of the system may be implemented partly as single issue and partly as dual issue. For example, the branch predict master block 220 works by loading and executing a straight-line block of instructions beginning with a particular PC, i.e., single issue. It then provides traces to the branch predict block which then unrolls as dual issue. Similarly, the iTags block 222 (which determines whether particular cache lines are in the icache) is implemented as single issue, while the icache is capable of reading two words out of the same cache line, i.e., dual issue.

[0041] In general, the interfacing of different issue circuitry enabled by the present invention allows the designer to implement particular circuitry as single or multiple issue depending upon particular design goals/constraints. For example, circuitry having certain complexities which make dual issue design difficult, e.g., state bits, may be implemented as single issue, while more straightforward circuitry may be implemented as dual or higher order issue. Referring back to Fig. 2, the issue order of the overall circuit ranges anywhere from ½ to 6 depending on conditions, so buffering may be added at various points, e.g., between the registers and backbone dispatch, to absorb fluctuations.

[0042] Fig. 3 shows an exemplary implementation of a dual issue to quad issue interface 300 according to a specific embodiment of the invention. The system in Fig. 3 shows a dual-issue resource 302 and a quad-issue resource 304 with a one-to-one mapping of data tokens between the two domains. The even pipe of resource 302 goes to the 0th and 2nd pipes of resource 304, and then the 0th and 2nd pipes of resource 304 returns to the even pipe of resource 302. The odd pipe of resource 302 goes the 1st and 3rd pipes of resource 304, and the 1st and 3rd pipes of resource 304 returns to the odd pipe of resource 302. This limited communication pattern makes the conversion circuits very simple. That is, according to a specific embodiment of the invention, the conversion circuitry employs alternating splits 306 on the send phase and alternating merges 308 on the return phase. This structure is useful in cases where the pipes associated with resource 304 may experience non-deterministic delay (or deterministic delay which is variable and difficult to design for), and the pipes on average maintain ½ the throughput of the pipes of resource 302. Thus by fanning out to quad-issue and then returning to dual-issue the overall system maintains the performance of the dual-issue pipe.

[0043] A practical example of this is in the "writeback" circuit of a standard RISC processor that implements precise exceptions. Precise exceptions are often time consuming to compute, and the writeback circuit can't write the results of the ALUs back into the register file until it knows that that instruction will not cause an exception. In this case the system of Fig. 3 can lead to improved performance.

[0044] To further generalize on the discussion above regarding alternative embodiments, Figs. 4A-4C show some generalized embodiments in which N-way-issue and M-way-issue resources interact in different ways. For example, Fig. 4A represents embodiments in which data are not returned to the resource from which they were received. Alternatively, Fig. 4B represents embodiments in which one resource acts as a data source for another. Finally, Fig. 4C represents embodiments in which data are both received from and returned to the originating resource. As discussed above, in embodiments corresponding to Fig. 4C, data may be received from and then returned to the originating resource in a more complex manner than a one-to-one correspondence with the appropriate logic and flow control.

[0045] Fig. 5 is an exemplary representation of more general embodiment of the invention. The system is divided into regions A, B, C. Region A contains a single multi-issue resource 502. Region C contains separate resources 504 and 506 that may be single or multi-issue. Region B contains circuitry 508 and 510 to facilitate the communication of data from the multi-issue resource in region A, to the resources in region C, and then to facilitate the communication of data from region C back to A, such that program order is maintained in both regions A and C. For the purposes of this disclosure, a pipe is defined as the single-issue component of which N in parallel, linked together by state processes flowing through neighboring pipes in sequence, constitute an N-way issue resource. Each pipe processes data in FIFO order. For example, if region A comprises one quad-issue resource, it has four pipes; and if region C comprises two dual-issue resources and one single issue resource, it has 5 pipes.

[0046] Consider the case where there is a one-to-one mapping of data tokens in the pipes in region A with data tokens in the pipes in region C for both communication going from A to C as well as the return path of C to A. In this case, an appropriate routing solution from region A to region C is a dispatch circuit. Generally speaking, a dispatch circuit is a circuit which is operable to route the data units received on a first number of input channels to designated ones of a second number of output channels in a deterministic manner thereby preserving a partial ordering for each output channel defined by a program order. In the context of the present invention, such a dispatch may be employed to connect an N-way issue resource to potentially multiple multi-issue resources such that ordering is preserved in each direction. An exemplary implementation of a dispatch circuit is described in U.S. Patent Publication No. US-2003-0146073-A1 for ASYNCHRONOUS CROSSBAR WITH DETERMINISTIC OR ARBITRATED CONTROL filed on April 30, 2002, the entire disclosure of which is incorporated herein by reference for all purposes. It will be understood that a variety of mechanisms may be used for this function within the scope of the invention.

[0047] Dispatch circuit 508 services the transfer of data of pipes in region A in order to the next available pipe in order out of all of the resources in region C. The routing decisions of dispatch circuit 508 are preserved and sent to return crossbar 510 that connects the pipes of region C back to the pipes of region A for the return path. For each region A pipe, the routing information identifying to which pipe in region C it's data was sent is the "merge control" of the return crossbar. The routing information of region C pipes identifying which region A pipe from which data are received is the "split control" of the return crossbar. As with the dispatch circuit, a crossbar for use with the present invention is described in U.S. Patent Publication No. US-2003-0146073-A1 incorporated herein by reference above.

[0048] As will be understood, there are many optimizations that can be made to the more general embodiment of Fig. 5 for particular applications. For example, the system of Fig. 1, a system that employs a dual-filter, dual-request, 1x2 split, and 2 2x1 optional assigns (or merges) is one such optimization. The system of Fig. 3, a system that employs alternating splits and alternating merges is another such optimization.

[0049] Another optimization is possible when the pipes in region C are fewer than the pipes in region A, e.g., because there is a low duty cycle requirement in region C. That is, not every piece of data needs to be processed in region C. In such a case, all of the data of region A is copied from the forward mapping to C, and used in the reverse mapping back to A. When data are computed in region C, it replaces the copy of this data during the reverse mapping back to region A. An optional assign (described above) may be used to do this.

[0050] While the invention has been particularly shown and described with reference to specific embodiments thereof, it will be understood by those skilled in the art that changes in the form and details of the disclosed embodiments may be made without departing from the spirit or scope of the invention. For example, the circuits and systems described herein may be represented (without limitation) in software (object code or machine code), in varying stages of compilation, as one or more netlists, in a simulation language, in a hardware description language, by a set of semiconductor processing masks, and as partially or completely realized semiconductor devices. The various alternatives for each of the foregoing as understood by those of skill in the art are also within the scope of the invention. For example, the various types of computer-readable media, software languages (e.g., Verilog, VHDL), simulatable representations (e.g., SPICE netlist), semiconductor processes (e.g., CMOS, GaAs, SiGe, etc.), and device types (e.g., processors, programmable logic devices, etc.) suitable for using in conjunction with the embodiments described herein are within the scope of the invention.

[0051] In addition, although various advantages, aspects, and objects of the present invention have been discussed herein with reference to various embodiments, it will be understood that the scope of the invention should not be limited by reference to such advantages, aspects, and objects. Rather, the scope of the invention should be determined with reference to the appended claims.